# IMPACT OF NUMA EFFECTS ON HIGH-SPEED NETWORKING WITH MULTI-OPTERON MACHINES

Stéphanie Moreaud and Brice Goglin

*INRIA – LaBRI – Université Bordeaux 1 – France*

{Stephanie.Moreaud,Brice.Goglin}@labri.fr

**ABSTRACT**

The ever-growing level of parallelism within the multi-core and multi-processor nodes in clusters leads to the generalization of distributed memory banks and busses with non-uniform access costs. These NUMA effects have been mostly studied in the context of threads scheduling and are known to have an influence on high-performance networking in clusters.

We present an evaluation of their impact on communication performance in multi-OPTERON machines. NUMA effects exhibit a strong and asymmetric impact on high-bandwidth communications while the impact on latency remains low.

We then describe the implementation of an automatic NUMA-aware placement strategy which achieves as good communication performance as a careful manual placement, and thus ensures performance portability by gathering hardware topology information and placing communicating tasks accordingly.

**KEY WORDS**

High-Speed Networks, NUMA, Opteron, Hypertransport, Placement, Performance.

## 1 Introduction

The emergence of clusters of workstations fifteen years ago led to the generalization of small SMP computing nodes in high-performance computing. This architecture is more generic and extensible, and less expensive than massive parallel supercomputers based on shared-memory. However, due to thermal dissipation problems, high-performance now requires increasing parallelization inside the nodes instead of high frequencies. When many CPU cores access the memory, the central bus becomes a major bottleneck and limits the scaling capabilities of the architecture. It is therefore now often replaced with a distributed architecture based on the interconnection of several processors and memory banks. These interconnects, such as AMD HYPERTRANSPORT [7], enable scalable assembly of multiple processor cores and memory banks into a large cache-coherent shared-memory multi-processor.

These architectures are becoming increasingly popular in high-performance computing where multiple large nodes are interconnected through one or several high-speed network interfaces such as INFINIBAND or MYRI-10G. Having the processors and memory banks physically distributed in the hardware leads to *Non-Uniform Memory Accesses* (NUMA). As data and tasks placement has an important impact on the overall local performance, the question of placement during network communication also has to be raised.

It is well known that networking benchmarks should be run with a careful NUMA-aware placement to exhibit optimal performance. Having to transfer data from a network interface to a distant processor or memory bank has an influence on the communication performance, either the latency or the bandwidth. Moreover, large nodes might have multiple network interfaces connected to different I/O busses making placement of multi-rail applications difficult.

We study in this article the impact of these NUMA effects on communication performance over high-speed networks. Our objective is firstly to measure these effects and exhibit situations where they become significant, and secondly to propose a way to automatically place communicating tasks and memory buffers to minimize these effects and maximize performance depending on the application needs. The rest of this paper is organized as follows. Section 2 describes the state of the art in the context of NUMA effects in large shared-memory nodes and high-performance networking in clusters. In Section 3, we detail the results of several micro-benchmarks that show the NUMA impact on low-level data transfer involving a network interface. Section 4 presents the corresponding observable effects on communications at the application level. Finally, we present in Section 5 how the middleware may automatically place communication buffers and tasks to reduce the influence of NUMA effects.

## 2 Background

### 2.1 Scheduling and Placement in NUMA Machines

Shared-memory multi-processors have been a successful architecture for a long time, possibly because they present a simple programming model, especially when compared strictly to non-shared-memory systems with explicit communication. Hardware performance is achieved thanks to the distribution of physical memory (to avoid the bottle-

necks of centralized memory busses) and the addition of caches (to decrease memory access time).

Placing and scheduling tasks becomes critical on such large systems since the affinity of the processes or threads with their caches or memory banks has a strong impact on the overall performance. The importance of placement decision increases substantially with the size and NUMA-ness of the system [4]. It led to several contributions on placement and scheduling of threads using locality and automatic page migration to maintain memory affinity [5].

Operating systems such as LINUX now expose NUMA-aware facilities [8] to the kernel subsystems and user-space applications so that memory affinities may be taken into account by the developers. User-space applications often use the corresponding `libnuma` library to place threads and memory according to their affinities. These facilities enable the accurate placement of data and threads, even with dynamic parallelism such as in OPENMP applications, but the application has to provide some hints to get the right placement and thus achieve the best performance.

### 2.2 AMD OPTERON and HYPERTRANSPORT

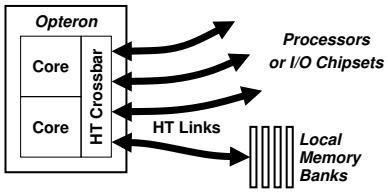


Figure 1. Architecture of the OPTERON processor with its own local memory banks and some HYPERTRANSPORT links to other NUMA nodes.

The emergence of the OPTERON processor in high-performance computing brings NUMA effects to all legacy MPI applications. While the common X86 architecture is centralized around a single memory bus, the AMD architecture distributes the processors and memory. Each OPTERON processor has its own memory banks attached to itself. They are still accessible to other processors, but the access time is increased (NUMA factor), hence making each processor its own NUMA node.

As described in Figure 1, all the nodes are connected altogether through the HYPERTRANSPORT bus [6], which is a cache-coherent interconnection network. All the cores of the OPTERON processor are connected to the local memory through a HYPERTRANSPORT crossbar. There might also be either 1 or 3 extra links, connecting up to 8 processors together. I/O chipsets are also connected to a dedicated link, making them closer to one NUMA node than the others. This architecture is known to have important NUMA effects on common applications in high performance computing [2].

We focus on multi-OPTERON machines in this article since this architecture exhibits interesting NUMA effects on high-speed networking due to the HYPERTRANSPORT topology, and also because it is the most commonly used NUMA architecture nowadays.

### 2.3 NUMA Effects on High-Speed Networking

The placement of communicating tasks is now often considered important. For instance, all high-speed networking benchmarks are placed manually on the processor and memory banks close to the network interface (using `numactl` tool) to reduce the latency and maximize the throughput. However, to the best of our knowledge, no extensive study of the NUMA impact in this case has been published so far. Yet, it is known that communication performance may be very sensitive to hardware characteristics and thus requires careful tuning.

On large systems, it has been shown that NUMA effects may have a strong impact on internal MPI communications within the machine. Cache sharing and appropriate memory placement improve communication performance between local tasks notably [1]. Still, there is very few support for NUMA affinities in existing message passing middlewares. For instance, the MX library (*Myrinet Express*) may only bind a single process per core to distribute the workload and make sure processes do not migrate. However, no indication about the actual underlying hierarchy of cores, processors and NUMA nodes is taken into account during this placement.

## 3 Micro-Benchmarks

### 3.1 Experimentation Platform

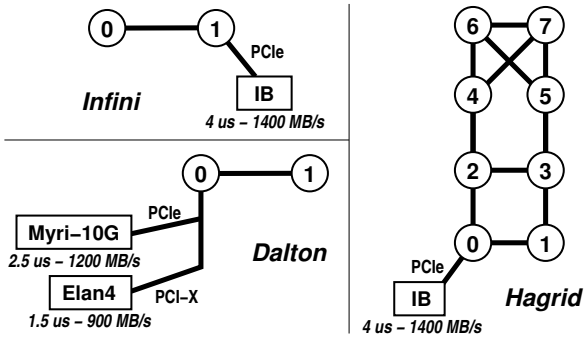


Figure 2. Topology of the experimentation platforms based on 2-socket and 8-socket dual-core OPTERON.

Our experimentation platform is made of several Dual-Core AMD OPTERON based machines, with either 2 or 8 sockets. Apart from the number of HYPERTRANSPORT links and their topology, the processor (1.8 GHz

dual-core OPTERON 265 and 865) and PCIE I/O chipsets (NVIDIA CK804) are identical.

As described in Figure 2, the I/O busses are connected to either the first or the second processor. The *Infini* machines are 2-socket hosts with a 8x INFINIBAND interface (PCIE Mellanox InfiniHost III) attached to the NUMA node #1, while *Hagrid* is a 8-socket host with the same interface on node #0. Depending on the benchmark, these machines achieve about 4 $\mu$s network latency and 1400 MB/s throughput.

The *Dalton* machines are 2-socket hosts with both a PCIE MYRICOM MYRI-10G and a PCI-X QUADRICS ELAN4 interfaces on node #0. The former network achieves about 2.5 $\mu$s latency and 1200 MB/s, while the latter exhibits 1.5 $\mu$s and 900 MB/s.

### 3.2 Latency

Achieving low-latency communication requires to reduce the critical path of data as much as possible. Thus, the best way to transfer small messages is often to write them by PIO (*Programmed Input/Output*) into the network interface (NIC) on the sender side, and have the NIC place them in the host memory by DMA (*Direct Memory Access*) on the receiver side. Even if this strategy may vary with the implementation, the expected influence of NUMA effects is always the same: the more HYPERTRANSPORT links between the sender task and its NIC (and between the remote NIC and the receiver task), the higher the latency should be.

| | Local Node | Distant Node | Overhead |
|---|---|---|---|
| MYRI-10G | 1739 | 1794 | 55 |
| ELAN4 | 1610 | 1670 | 60 |
| INFINIBAND | 464 | 559 | 95 |

Table 1. Impact of NUMA placement on small request round-trip latency (in nanoseconds) with various high-speed interconnect interfaces on 2-socket machines.

Table 1 presents the delay between issuing a command by PIO write to a NIC and the arrival of the completion event (either by PIO or by DMA depending on the hardware). We obtained INFINIBAND results with a manual PIO write followed by a PIO read. MYRI-10G and ELAN4 results are based on similar dedicated benchmarks provided by their vendors (`mx_piobench` and `qsnet2_dmatest`)[1].

When comparing a run on a NUMA node close and distant from the NIC, we observed between 55 and 95 nanoseconds of overhead for this round-trip from one processor to the NIC, which means one HYPERTRANSPORT one-way hop costs about 40 ns. We extended this experiment to our 8-socket machine and confirmed that the one-way latency increases by about 40 ns with each additional HYPERTRANSPORT hop.

### 3.3 Bandwidth

We now present the impact of NUMA placement on bandwidth by looking at local DMA performance. Table 2 presents the transfer bandwidth depending on the copy initiator and memory location. It first shows that the bigger the optimal bandwidth is, the bigger performance drop is observed when the placement is distant. Indeed, ELAN4 does not expose any variation while INFINIBAND and MYRI-10G suffer notably. Also, DMA read does not appear to suffer from the NUMA effect, only DMA write does. Comparing to NUMA impact on memory copies confirms that the performance drop increases with the theoretical bandwidth, and that writing suffers more than reading.

| | | Local | Distant | Impact |
|---|---|---|---|---|
| ELAN4 DMA | Read | 879 | 879 | 0 % |
| | Write | 925 | 925 | 0 % |
| INFINIBAND | Read | *1075* | *1075* | 0 % |
| DMA[2] | Write | *1392* | *1071* | -23 % |
| MYRI-10G | Read | 1308 | 1295 | 1 % |
| DMA | Write | 1518 | 1162 | -24 % |
| Processor | Read | 2696 | 1871 | -31 % |
| Memory Access | Write | 4765 | 2952 | -38 % |

Table 2. Impact of NUMA placement on memory bandwidth (in MB/s) on 2-socket hosts, when initiating data transfer from a NIC (DMA) or from a processor.

## 4 Application-Level Benchmarks

### 4.1 Latency

Observing the communication latency on the MYRI-10G network reveals an almost negligible influence of the NUMA placement. Only a 25 ns overhead appears per distant placement while the base latency is about 2.5 $\mu$s, which means the whole variation is about 2 % on 2-socket hosts. It is even more negligible on INFINIBAND since its base latency is much higher.

The ELAN4 network shows a higher influence since its base latency is much lower, between 1 and 2 $\mu$s depending on the communication mode. When using *Put/Get* operations, the raw latency is very close to 1 $\mu$s and the impact of a distant placement reaches 100 ns on both sides, which means an overall 20 % latency difference. This high impact

[1] These tools are very specific and should not be compared to each other from row to row, but they provide a similar way to evaluate the impact of NUMA placement on latency.

[2] Based on actual network communication instead of a local DMA benchmark, and hence lower than the theoretical DMA capability of the hardware.

might be related to ELAN4 using a special way to transfer data on the PCI-X and HYPERTRANSPORT busses from the host to the NIC, for instance with different PCI packet sizes.

In the case of large machines such as *Hagrid* and such communication modes, the NUMA impact on latency might thus become important for latency sensitive applications. However, for other networks or communication modes on common machines, the NUMA influence is merely negligible.

### 4.2 Bandwidth

Figure 3 shows the impact of NUMA placement on an extreme usage of network bandwidth, a multi-rail ping-pong with the NEWMADELEINE middleware [3]. Each application message is transparently stripped across both networks depending on their respective performance. When the message size reaches 32 kilobytes, the distant placement starts to limit the performance, with up to 40 % difference between the distant placement and the optimal throughput (almost 2 GB/s for 8 MB messages). We observed similar effects on various communication modes and networks.

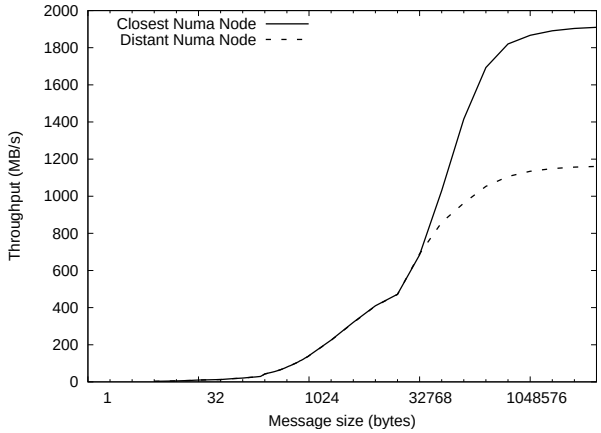


Figure 3. Performance of multi-rail ping-pong with NEWMADELEINE over ELAN4 and MYRI-10G networks depending on the placement of memory buffers on both 2-socket machines.

When placed on a distant NUMA node, the aggregate bandwidth is even lower than single-rail MYRI-10G ping-pong. It might be caused by congestion related to two I/O chipsets being used simultaneously in the multi-rail case since they are connected to the same HYPERTRANSPORT link on NUMA node #0. We observed that, when reaching about 1 GB/s, between 40 and 60 % of any additional throughput is lost in case of distant placement. The performance drop seems to be higher in multi-rail tests, likely because of congestion caused by the traversal of two I/O chipsets connected to the same NUMA node.

### 4.3 Asymmetric Effects

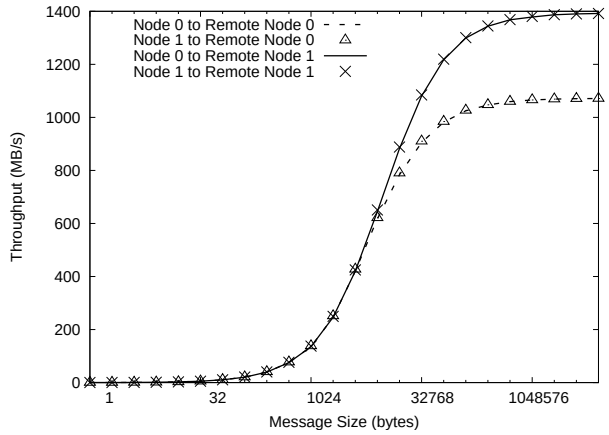


Figure 4. Performance of RDMA Write with OPENIB VERBS on 2-socket machines depending on the placement on both machines.

Figure 4 presents the impact of NUMA placement on the throughput of a RDMA write stream over INFINIBAND. It shows the same behavior than the previous multi-rail ping-pong, with a saturation before 1100 MB/s, 25 % below the performance of the optimal placement.

However, this figure also enables comparison of the NUMA effect on the sender and receiver sides since this test is not bidirectional as the previous multi-rail ping-pong. It shows that the placement of the target buffer of the RDMA write is important while the sender is not. We actually observed the same behavior on send/receive and RDMA read communication: only the location of the buffer where data will be *written* is important. The HYPERTRANSPORT specifications [6] lead us to think that this surprisingly asymmetric effect might be caused by a saturation of the bus related to the asymmetric numbers of request and response buffers in the hardware. Although some tuning might be possible through the BIOS, it does not look like a feasible solution for production systems for now.

Figure 5 extends this result to our 8-socket machine and shows that the RDMA write throughput does not decrease when additional HYPERTRANSPORT hops have to be traversed. Once the placement is distant, increasing the actual distance between the NUMA node and the NIC does not reduce the performance further.

## 5 Automatic Configuration

Networking benchmarks are always placed by hand with `numactl` to exhibit optimal performance, but also to make it reproducible. Failing to bind a task to a processor leads to random migration by the scheduler, for instance when a background daemon wakes up.

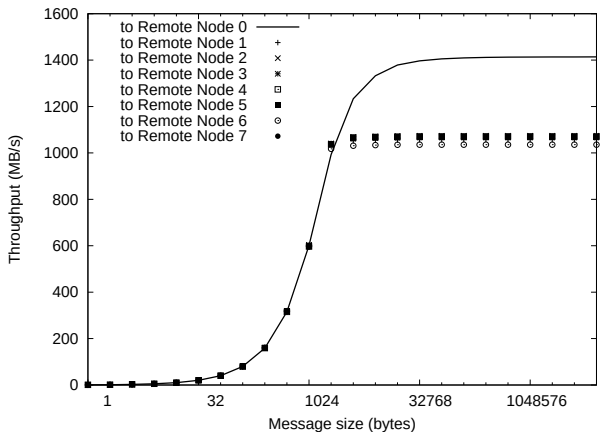


Figure 5. Performance of RDMA Write with OPENIB VERBS depending on the receiver location on a 8-socket machines.

In the general case of parallel applications, the number of tasks per machine is usually equal to the number of cores. Unless some tasks use the network more than the others, the NUMA location of the network interfaces should not have any visible major influence on the overall performance. However, in the case of irregular network needs among the tasks on the same node, the placement might be important since those closer to the interfaces will have faster access to the network.

Performance portability should be guaranteed by the operating system or runtime middleware by placing tasks and buffers automatically. Any user deploying an application on a new cluster should not have to look into the motherboard details to find out where the network interfaces are attached. Therefore, it is interesting to provide an automatic placement engine. This is what we are presenting in this section, by first looking at the placement of a single process, and then at the global placement of multiple tasks.

### 5.1 Determining distances between nodes and an I/O device

Placing tasks and buffers near a network interface first requires to know which NUMA node is physically close to it. Micro-benchmarks such as those used in Table 1 help detecting of the physical location of a board, but the availability and reliability of these tools may not be sufficient for production use. Exposing the NUMA topology to the middleware appears to be a better solution. The operating system knows which NUMA node each PCI device is attached to. We integrated a patch in LINUX 2.6.21 to expose the NUMA node each physical device is attached to, and thus make it easy for middlewares to use `libnuma` for task and buffer binding depending on I/O device locations.

However, these attributes cannot be immediately translated into exploitable placement information at the application level since the application manipulates virtual handles (for instance a MX endpoint, a VERBS queue pair, or a socket) pointing to physical devices. A generic interface to expose NUMA attributes also appears difficult to define since all interconnects do not manipulate the same type of virtual handles. An interesting solution consists in letting the middleware take care of this problem inside its existing virtualization layer. We implemented this idea in the NEWMADELEINE middleware which is developed in our research team [3]. We added to NEWMADELEINE drivers for INFINIBAND and MYRI-10G networks a way to retrieve the NUMA location of the physical devices involved in application-level communication.

### 5.2 Automatic Placement

Once the NEWMADELEINE middleware obtained the physical location of the underlying network interfaces, it has to place the buffers and tasks accordingly. It is important to place at the right time during the execution since an early placement could restrict unrelated resources (for instance other computing threads that the middleware might deploy on all the cores) while a late placement might miss some critical resource allocations (for instance some pages that have been pinned down in physical memory for DMA from/to the NIC). The middleware therefore needs to query each driver for NUMA attributes, then place accordingly, and finally initialize the drivers.

We implemented this strategy in NEWMADELEINE and verified that it achieves as good performance as careful manual placement. Any application using MYRI-10G or INFINIBAND networks is automatically placed on the NUMA node close to the interfaces. When another interconnect is used without providing NUMA attributes (for instance ELAN4) NEWMADELEINE assumes that no NUMA node is closer to the NIC than the others, and thus does not take this NIC into account while making placement decision.

### 5.3 Multi-Rail Placement

Multi-rail applications need to be placed carefully since their performance expectation is high and all network interfaces that are involved may not expose the same NUMA affinities. It is common nowadays to have 2 I/O busses attached to different NUMA nodes on OPTERON machines.

In the case of heterogeneous multi-rail, the middleware has to compare the NUMA attributes of each interconnect and take their performance into account before making any placement decision. Indeed, a latency bounded application might be better placed near a QUADRICS network while a bandwidth bounded should probably be close to an INFINIBAND network. Also, a CPU bounded application would rather be placed on the available processors than on a busy processor near the network interface. Therefore, the application should provide placement hints depending on its needs.

In the common case of homogeneous multi-rails with similar interconnects connected to different NUMA nodes, no placement will be optimal. Interleaved memory allocation might help since it distributes the workload on the memory banks and thus reduces the risk of congestion.

## 6 Conclusion and Future Works

This article presents a study of the NUMA effects on high-speed networking in clusters of multi-OPTERON machines These effects are well known in the context of scheduling and placement of threads and data in a single computation host. However, to the best of our knowledge, no extensive study in the case of distributed computing with high-speed network communication has been published. Contrary to the traditional centralized bus, the topology of the HYPERTRANSPORT bus within the OPTERON architecture places the I/O devices close to a single NUMA node.

We presented micro-benchmarks and application-level measurements showing that placing a task on a NUMA node far from the network interface leads to a performance drop. While the latency only slightly increases with the distance between the components in the machine (about 40 ns per HYPERTRANSPORT hop), the throughput exhibits a dramatic decrease for bandwidth intensive transfers, up to 40 % at 2 GB/s. We also showed that this NUMA effect on the throughput is actually asymmetric since only the target destination buffer appears to need a placement on a NUMA node close to the interface. This phenomenon might be caused by the configuration of hardware buffers in the HYPERTRANSPORT chipsets.

Once the NUMA impact on high-performance networking was evaluated, we presented an implementation of automatic placement in the NEWMADELEINE communication middleware on top of INFINIBAND and MYRI-10G interconnects. We exposed the relevant NUMA attributes of the network devices to the application and used them to place single tasks close to these devices. This implementation enables performance portability by automatically correctly placing single tasks to achieve as good communication performance as in case of manual placement, even for multi-rail networking.

We now plan to look more deeply at the impact of NUMA effects on the existing communication modes of various interconnects. PIO, copy+DMA and memory registration+DMA have different requirements on the placement of tasks and buffers. Indeed, the processor is not involved during DMA while the memory is only slightly involved in PIO. The application should be able to provide placement hints to help the middleware when choosing between supporting latency, throughput or CPU availability. Also, the congestion on the memory bus has to be studied since its effect on a loaded machine is expected to be important. Finally, we are also looking at other NUMA architectures, especially ITANIUM 2 based machines with a hierarchical memory bus, or the upcoming INTEL QUICKPATH.

These pieces of information should enable a clever placement of applications mixing OPENMP parallel sections (which distribute a lot of threads across the machines), and MPI communications (which involve a small amount of threads that should be placed near the corresponding I/O devices). Integrating our NUMA information in a thread scheduler will make it easy to bind communicating threads closer from the network.

## References


[1] Sadaf R. Alam, Richard F. Barrett, Jeffery A. Kuehn, Philip C. Roth, and Jeffrey S. Vetter. Characterization of Scientific Workloads on Systems with Multi-Core Processors. In *Proceedings of the IEEE International Symposium on Workload Characterization (IISWC)*, San Jose, CA, 2006.

[2] Joseph Antony, Pete P. Janes, and Alistair P. Rendell. Exploring Thread and Memory Placement on NUMA Architectures: Solaris and Linux, UltraSPARC/FirePlane and Opteron/HyperTransport. In *Proceedings of the International Conference on High Performance Computing (HiPC)*, Bangalore, India, December 2006.

[3] Olivier Aumage, Elisabeth Brunet, Guillaume Mercier, and Raymond Namyst. High-Performance Multi-Rail Support with the NewMadeleine Communication Library. In *Proceedings of the Sixteenth International Heterogeneity in Computing Workshop (HCW 2007), held in conjunction with IPDPS 2007*, Long Beach, CA, March 2007.

[4] Timothy Brecht. On the Importance of Parallel Application Placement in NUMA Multiprocessors. In *Proceedings of the Fourth Symposium on Experiences with Distributed and Multiprocessor Systems (SEDMS IV)*, San Diego, CA, Sept 93.

[5] Rohit Chandra, Scott Devine, Ben Verghese, Anoop Gupta, and Mendel Rosenblum. Scheduling and page migration for multiprocessor compute servers. In *Proceedings of the sixth international conference on Architectural support for programming languages and operating systems table of contents*, pages 12–24, San Jose, CA, 1994.

[6] HyperTransport I/O Link Specification. `http://www.hypertransport.org/`.

[7] Chetana N. Keltcher, Kevin J. McGrath, Ardsher Ahmed, and Pat Conway. The AMD Opteron Processor for Multiprocessor Servers. *IEEE Micro*, 23(2):66–76, March 2003.

[8] Christoph Lameter. Local and Remote Memory: Memory in a Linux/NUMA System, July 2006.